H0004992US

ENSURE A CONSISTENT CONTROL SYSTEM CONFIGURATION METHODOLOGY THROUGH AN ENFORCEABLE USER DEFINED DEVELOPMENT LIFE CYCLE

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0001] The present disclosure generally relates to source control systems and development life cycles. In particular, the present disclosure relates to regulated industry, development, qualification, process automation, and other applications and features.

2. Discussion of the Background Art

[0002] A systems development life cycle (SDLC) model is also called the life cycle or the waterfall model. The life cycle model is an approach to developing an information system or software product in a sequence of steps that progress from start to finish. Typically the steps include system requirements, software requirements, analysis, program design, coding, testing, and maintenance. The life cycle model is one of the oldest systems development models and is still probably the most commonly used.

[0003] Source control is also known as configuration management and change management. Source control is a discipline of making changes to source code in a planned and systematic fashion. The purpose of source control is to formally control the integrity of artifacts (items) and activities (tasks). In a source control system, objects are checked-out, edited, and then checked-in. Each time

an object is checked-in, it is given a version number. Over time, a history of changes is created for the objects under the control of the source control system.

[0004] The items under control in a source control system includes objects, such as control strategies. In object-oriented programming (OOP), objects are abstractions used in designing a program and they are also the units of code that are eventually derived from the design process. In between, each object is made into a generic class of objects and even more generic classes are defined so that objects can share models and reuse the class definitions in their code. Each object is an instance of a particular class or subclass with the class's own methods or procedures and data variables. Thus, objects typically exist in a hierarchy of objects with parent and child relationships. An object is usually a binary, text, or other type of file.

[0005] Process control systems are used to control and monitor complex processes in many types of industrial settings, including refineries, pharmaceuticals, power and chemical plants, and pulp, paper and printing mills. One example is Experion PKS™ available from Honeywell.

[0006] Many industries need to comply with regulations from federal agencies, such as the Food and Drug Administration (FDA) or the Environmental Protection Agency (EPA). In many cases, the overall production process, including a process automation system, needs to be validated. For the process automation system, this means that development needs to be performed in a consistent manner following standard operating procedures, maintaining a change history of configuration control. This is a complicated and time-consuming effort and there is a need for it to be automated and enforced. This would reduce the work and cost associated with the overall validation effort.

[0007] There is a need for a way for a user to define and use a development life cycle as part of a configuration tool in a process automation system. The configuration tool would automate and enforce the development life cycle for each individual control object through user predefined qualification states. The developer would only be able to make predefined qualification state transitions to which he or she was granted access. The sign-off in typical standard operating procedures is performed manually on printed-paper. There is a need for automating the sign-off and logging each interaction with the developers, such as the action performed and the time and date. This would prevent the control objects that are not yet intended to be part of the production process from being loaded. This would also prevent important development steps like testing and validation from being skipped.

SUMMARY OF THE INVENTION

[0008] The present invention has many aspects and is generally directed to ensuring a consistent control system configuration methodology through an enforceable user defined development life cycle.

[0009] One aspect is a method for enforcing a life cycle process in a source control system. A user-defined life cycle process is received having a plurality of states, where each state has attributes. User-defined state transitions between the states are received. A change state function is provided for changing a current state associated with an object to a next state associated with the object, where the change state function verifies compliance with the user-defined state transitions. Version control is provided for the object in the source control system.

[0010] Another aspect is a computer readable medium, such as a floppy disk, a website or the like having executable instructions stored thereon to perform a method of determining permissions for actions with an object based on a state of the object. The instructions are executable on a processor. A request to perform an action with the object is received. It is determined whether the object has ever been checked-in to the source control system. It is determined whether the object is currently checked-in. A definition of the state of the object is retrieved. It is determined from the definition whether the action is permissible in the state. A permission status is provided.

[0011] Another aspect is a computer readable medium having executable instructions stored thereon to perform a method of validating state transitions. A request to make a state transition for an object from a user is received. It is determined whether the object is checked-in. It is determined whether the user has permission to make the state transition based on a user-defined state transition model. The state transition is permitted only if the user has permission. The state transition status is provided.

[0012] Another aspect is a computer readable medium having executable instructions stored thereon to perform a method of validating a state transition. It is determined whether a next state in a state transition request from a user is allowed from a current state in the state transition request based on user-defined transition restrictions. It is determined whether the user has permission to make the state transition based on user-defined transition restrictions. A state transition status is provided.

[0013] Another aspect is a computer readable medium having executable instructions stored thereon to perform a method of validating a state transition. It is determined whether the current state transition in a state transition

request for an object from a user requires an electronic signature based on user-defined transition restrictions. It is determined whether a previous state transition for the object required a previous electronic signature, if the current state transition requires a current electronic signature. The current state transition is allowed only if the previous electronic signature is different than the current electronic signature. A validation status is provided.

[0014] Another aspect is a computer readable medium having executable instructions stored thereon to perform a method of determining a new state for an object version upon check-in. It is determined whether the object is being checked-in for the first time. A first fallback state for a first pre-defined state is retrieved, if the object is being checked-in for the first time. The first fallback state is provided, if the object is being checked-in for the first time.

[0015] Another aspect is a computer readable medium having executable instructions stored thereon to perform a method of processing the addition of a state. A definition of a new state is received from a user, where the definition includes a name and a fallback state. It is determined whether the name is unique among existing state definitions. The fallback state is validated. The definition is added to a source control system, only if the name is unique and the fallback state is valid.

[0016] Another aspect is a computer readable medium having executable instructions stored thereon to perform a method of processing the modification of a state. A modified definition of a state is received from a user, where the modified definition includes a name and a fallback state. It is determined whether the name is unique among existing state definitions. The fallback state is validated. The modified definition is updated in a source control system, only if the name is unique and the fallback state is valid.

[0017] Another aspect is a computer readable medium having executable instructions stored thereon to perform a method of processing the deletion of a state. A request to delete a state definition for the state is received from a user. It is determined whether the state definition is referenced by any other state definition in a source control system. It is determined whether any objects in the source control system have a current state equal to the state. The state definition is deleted from the source control system, only if the state definition is not referenced by any other state definition in the source control system and no objects in the source control system have the current state equal to the state.

[0018] Another aspect is a source control system for a process control system which comprises a processor, a life cycle process component, a version control component, and a controller. The life cycle process component is executable on the processor to enforce compliance with user-defined life cycle states. The version control component is executable on the processor to associate a version number with each object. The controller is in communication with the processor via a network to be loaded with the objects to provide process control for a plurality of devices.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] These and other features, aspects, and advantages of the present disclosure will become better understood with reference to the following description, appended claims, and drawings where:

[0020] FIG. 1 is a flow chart of an example life cycle in a source control system;

[0021] FIG. 2 is a screenshot of an example user interface for a qualification life cycle configuration;

[0022] FIG. 3 is a screenshot of an example user interface for adding a new qualification state;

[0023] FIG. 4 is a screenshot of an example user interface for modifying a qualification state;

[0024] FIG. 5 is a screenshot of an example user interface for deleting a qualification state;

[0025] FIG. 6 is a screenshot of an example user interface for configuring state transition requirements;

[0026] FIGS. 7A and 7B are screenshots of an example user interface for changing a qualification state;

[0027] FIG. 8 is a screenshot of an example user interface for entering an electronic signature;

[0028] FIG. 9 is a screenshot of an example user interface for configuring electronic signatures;

[0029] FIG. 10 is a flow chart of an example method of determining permission for an action with an object based on its qualification state;

[0030] FIG. 11 is a flow chart of an example method of validating a qualification state transition;

[0031] FIG. 12 is a flow chart of an example method of validating a qualification state transition based on transition restrictions;

[0032] FIG. 13 is a flow chart of an example method of validating a qualification state transition based on restricted signing;

[0033] FIG. 14 is a flow chart of an example method of determining the qualification state for an object version upon check-in;

[0034] FIG. 15 is a flow chart of an example method of processing the addition of a qualification state;

[0035] FIG. 16 is a flow chart of an example method of processing the modification of a qualification state;

[0036] FIG. 17 is a flow chart of an example method of processing the deleting of a qualification state;

[0037] FIG. 18 is a block diagram of an example system architecture for a source control system having an enforceable user-defined life cycle; and

[0038] FIG. 19 is a block diagram of another example system architecture for a source control system having an enforceable user-defined life cycle.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0039] FIG. 1 shows an example life cycle in a source control system. In this example, the life cycle is defined by the qualification states {implemented, tested, decommissioned, validated}. First, a control strategy is created 102. Then, the control strategy is checked-in to the source control system 104 and its qualification state is set to "implemented". After it is checked-in, the control strategy is loaded to a controller and tested 106. If the control strategy does not pass the test 108, it is checked-out and modified 110 and, then, checked-in again 104. Once the control strategy passes the test 108, its qualification state is updated to "tested" 112. Next, the control strategy is validated. If the control strategy does not pass validation 114, its qualification state is set to "decommissioned" 116; otherwise, its qualification state is set to "validated". The present invention includes a method for users to track life cycle phases or states, such as the sequence of qualification states for control strategies in the above example. States may be any type of state associated with a life cycle process, such as a qualification state. The present invention is not limited to particular life cycle phases or qualification states.

[0040] FIG. 2 shows an example user interface for a qualification life cycle configuration. In this example, qualification state refers to the stages a control strategy passes through as part of the customer's qualification process for the control strategy. For example, a customer may define {implemented, tested, released, withdrawn} as states for tracking the process. The control strategy object goes through these different states as it is developed and refined. In some cases, the control strategy goes through certain states in order to be qualified. For example, the control strategy needs to be implemented and tested before it is released.

[0041] In FIG. 2, qualification life cycle states 200 and qualification life cycle transitions 202 are configurable by a user. Add 204, delete 206, and modify 208 functions are provided for qualification life cycle states 200 and a configure state transition requirements function 210 is provided for qualification life cycle transitions 202. In this example, the user has defined {implemented, testing, released, withdrawn} as the qualification states 212. Each qualification state is assigned a fallback state 214. Each qualification state has a value for whether there is restricted signing 216, and whether loading to the controller is allowed 218. In this example, the user has defined state transitions in a grid 219 with columns "to states" 220 and rows "from states" 222 for qualification life cycle transitions 202. In this example, some state transitions are grayed out (e.g., implemented to implemented) because they are not possible, some state transitions are not allowed, and others are assigned users or groups of users, who are allowed to make the transition. Grid 219 shows all the qualification states defined in the first row and first column of grid 219. The user has an option of configuring electronic signatures for state transitions. For example, if the user wants to configure electronic signature for the transition from implemented to testing, then the user chooses the cell with the row corresponding to implemented and the

column corresponding to testing and clicks on configure transition requirements 210.

[0042] FIG. 3 shows an example user interface for adding a new qualification state. This user interface is displayed when the user selects add 204 from the user interface in FIG. 2 and it prompts the user to enter a new qualification state 300, a fallback state 302, whether there is restricted signing 304, and whether load to controller is allowed 306.

[0043] FIG. 4 shows an example user interface for modifying a qualification state. This user interface is displayed when the user selects modify 208 from the user interface in FIG. 2 and it shows the user the name of the qualification state being modified 400, its current fallback state 402, whether there is restricted signing 404, and whether load to controller is allowed 406. These fields are editable.

[0044] FIG. 5 shows an example user interface for deleting a qualification state. This user interface is displayed when the user selects delete 206 from the user interface in FIG. 2 and it confirms the delete action.

[0045] FIG. 6 shows an example user interface for configuring state transition requirements. This user interface is displayed when the user selects a cell in grid 219 and clicks on configure transition requirements 210 from the user interface in FIG. 2. The current state transition requirement values are displayed 600 and the user chooses whether there is a signature requirement for the state transition 602. "Not allowed" 604 indicates that the state transition is not allowed between the states. "Anybody" 606 indicates that any user can change transition state. "User ID" 608 indicates that only the specified user 610 (chosen from a list) can change transition state. "Group" 612 indicates that only the users belonging to

a specified group 614 (chosen from a list) can change transition state. Checking the electronic signature requirement 602 prompts the user for an electronic signature when the state transition occurs.

[0046] FIGS. 7A and 7B shows an example user interface for changing a qualification state. As part of qualification state transitions, this user interface is invoked to accommodate transitioning from one state to another. The object 700, its current version 702, and qualification state 704 are displayed with an option to change the qualification state to another pre-defined value 706. Only allowed state transitions are shown in the drop down menu as shown in FIG. 7B.

[0047] FIG. 8 shows an example user interface for entering an electronic signature. For some qualification state transitions, the source control system is configured by the user to require verification by validating an electronic signature. To validate the electronic signature, an electronic signature component is invoked. The user name 800 and password 802 are entered before a time-out period expires.

[0048] FIG. 9 shows an example user interface for configuring electronic signatures. The timeout period and the number retries by the user is configurable by the user.

[0049] In an example source control system, the number and naming of the qualification states is configurable by the customer. Each version of each object in the source control system is associated with a qualification state. In this way, a procedure for qualification of control strategies is enforceable. Each qualification state has an attribute indicating if it is permissible to load the object to a controller. Each qualification state is defined with a set of other qualification states to which that state may transition. Each qualification state is defined with

certain users or groups of users who are authorized to make those transitions. A transition may be associated with an electronic signature requirement. Each qualification state has a fallback state. A fallback state is the state that an object is placed in when it is checked-in to the source control system, after it has been checked-out from the source control system. The first time the object is checked-in the source control system, its qualification state is set to the fallback state of the first defined state. A transition from one qualification state to another is a user-specified action. The qualification state of an object is automatically changed, if necessary, at the time it is checked-into the source control system. Also, the state of a checked-in object is changeable at any time without being part of the check-in. A log entry is generated when the state is changed, including a name of the user, the state the object changed from, and the state the object changed to.

[0050] In the example source control system, there is a restricted signing state for each qualification state. If a restricted signing state requires an electronic signature and the state to which a transition is being made also requires an electronic signature, then the two signatures need to be different. A user may specify that a state does not require a restricted signing state. This embodies the Good Automated Manufacturing Practice (GAMP) "Four Eyes" principle. For more information, see "The Good Automated Manufacturing Practice (GAMP) Guide for Validation of Automated Systems in Pharmaceutical Manufacture," available from The Society for Life Science Professionals (http://www.ispe.org). Changing the qualification state of an object version does not change the version number. A default installation of the source control system includes a database with a single qualification state named "implemented" with an attribute set to allow loading to a controller and a fallback state set to "implemented".

[0051] In the example source control system, a user interface is available only to a user having a privilege for configuring qualification states,

including the following information: number of states, name of each state, a "load to controller allowed" attribute for each state, a restricted signing state for each state, permitted state transitions, whether authorization is needed for each transition for a specific user or a user who belongs to a specified group, whether electronic signatures are needed for each transition, and a fallback state. In the example source control system, configuration is available in administrative functions.

[0052] FIG. 10 shows an example method of determining permission for an action with an object based on its qualification state. Security is described in this example, but is optional in the present invention and may be performed in many different ways. In step 1000, a specific action is requested for an object. In step 1002, it is determined whether the object has ever been checked-in to the source control system. If not, control flows to step 1004, where the permission status is set to okay and, then, control flows to step 1006, where the permission status is returned. Status setting and other error processing are shown in this example and other flow charts, but may be omitted or done in many different ways. In step 1002, if the object has been checked-in before, then control flows to step 1008. In step 1008, it is determined if the object is currently checked-in. If not, control flows to step 1010, where the permission status is set to indicate that the action cannot be performed and, then, control flows to step 1006. Otherwise, if the object is currently checked-in, control flows to step 1012. In step 1012, a definition of the qualification state of the object is retrieved. In step 1014, it is determined if the qualification state permits the action requested in step 1000. If not, control flows to step 1016, where the permission status is set to indicate that the action cannot be performed and, then, control flows to step 1006. Otherwise, if the qualification state permits the action, then control flows to step 1004.

[0053] FIG. 11 shows an example method of validating a qualification state transition. In step 1100, the user requests a qualification state transition for an object. In step 1102, it is determined whether the user has permission to make a qualification state transition of any kind. If not, control flows to step 1104, where a status is set to indicate that the qualification state transition cannot be performed and, then, control flows to step 1106, where the processing for the request to update the qualification state terminates. Otherwise, if the user has permission, control flows to step 1108. In step 1008, it is determined whether the object is checked-in. If not, control flows to step 1110, where the status is set to indicate that the qualification state transition cannot be performed and, then, control flows to step 1106. Otherwise, if the object is checked in, control flows to step 1112. In step 1112, it is determined if the object can be locked in a source control database. The source control database is any type of storage device capable of storing information about objects in the source control system, such as version numbers. If the object cannot be locked, control flows to step 1114, where the status is set to indicate that the qualification state transition cannot be performed and, then, control flows to step 1106. Otherwise, if the object can be locked, control flows to step 1116. In step 1116, it is determined whether the user has permission to make this specific qualification state transition. (See FIG. 12 for a method of validating a qualification state transition based on transition restrictions). If not, control flows to step 1118, where the status is set to indicate that the qualification state transition cannot be performed and, then, control flows to step 1106. Otherwise, if the user has permission, control flows to step 1120, where the qualification state is updated for the object version and the object is unlocked. Then, control flows to step 1106 and the request processing terminates.

[0054] FIG. 12 shows an example method of validating a qualification state transition based on transition restrictions. In step 1200, a status is set to

success. In step 1202, it is determined whether the qualification state requested is allowed from the current state. If not, control flows to step 1204, where the status is set to indicate that the qualification state transition cannot be performed and, then, control flows to step 1206, where the status is returned to the user. Otherwise, if the transition is allowed from the current state, control flows to step 1208. In step 1208, it is determined whether the requested qualification state has a restricted signing state. If so, control flows to step 1210, where it is determined whether the criteria for is met restricted signing for the state transition. If not, control flows to step 1206. Otherwise, control flows to step 1212. In step 1212, it is determined whether the user is in the list of allowed users to make this transition. If not, control flows to step 1214, where the status is set to indicate that the qualification state transition cannot be performed and, then, control flows to step 1206. Otherwise, if the user is in the list, control flows to step 1206.

[0055] FIG. 13 shows an example method of validating a qualification state transition based on restricted signing. In step 1300, the status is set to success. In step 1302, it is determined whether the transition from the current state to the requested qualification state requires an electronic signature. If not, control flows to step 1304, where the status is returned. Otherwise, if it requires an electronic signature, control flows to step 1306. In step 1306, it is determined whether any transition to the qualification state that is the restricted state requires an electronic signature. If not, control flows to step 1304. Otherwise, control flows to step 1308. In step 1308, it is determined whether the electronic signatures are different for the current transition and the previous transition to the restricted qualification state. If so, control flows to step 1304. If not, control flows to step 1310, where the status is set to indicate that the qualification state transition cannot be performed and, then, control flows to step 1304.

[0056] FIG. 14 shows an example method of determining the qualification state for an object version upon check-in. In step 1400, it is determined whether this is the first time the object has been checked-in to the source control system. If so, control flows to step 1402, where the fallback state for the first defined qualification state is retrieved and, then, control flows to step 1406, the end of the qualification state determination. Otherwise, if it is not the first time, control flows to step 1408. In step 1408, the qualification state is retrieved for the previous version of the object. In step 1410, the fallback state is retrieved for the retrieved qualification state.

[0057] FIG. 15 shows an example method of processing the addition of a qualification state. In step 1500, receiving a definition of a new qualification state from a user through a user interface. In step 1502, it is determined whether the user has the privilege to update the qualification state definitions. If not, control flows to step 1504, where an error status is set and, then, control flows to step 1506, where the status is returned to the user. Otherwise, if the user has the privilege, control flows to step 1508. In step 1508, it is determined whether the locking of the qualification state definitions was successful. If not, control flows to step 1510, where an error status is set and, then, control flows to step 1506. Otherwise, if the locking was successful, control flows to step 1512. In step 1512, it is determined whether the name of the qualification state being added is unique, i.e., not used by other qualification states. If not, control flows to step 1514, where an error status is set and, then, control flows to step 1515. In step 1515, the qualification state definitions are unlocked and, then, control flows to step 1506. Otherwise, if the name is unique, control flows to step 1516. In step 1516, it is determined whether there is a valid fallback qualification state specified. If not, control flows to step 1518, where an error status is set and, then, control flows to step 1515. Otherwise, if there is a valid fallback qualification state, control flows

to step 1520. In step 1520, it is determined whether a valid restricted signing qualification state is specified. If not, control flows to step 1522, where an error status is set and, then, control flows to step 1515. Otherwise, if a valid restricted signing qualification state is specified, control flows to step 1524. In step 1524, it is determined whether the qualification state was successfully added to the database. If not, control flows to step 1522, where an error status is set and, then, control flows to step 1515. Otherwise, if it was successfully added, control flows to step 1515.

[0058] FIG. 16 shows an example method of processing the modification of a qualification state. In step 1600, receiving a definition of an updated qualification state from the user through a user interface. In step 1602, it is determined whether the user has the privilege to update the qualification state definitions. If not, control flows to step 1604, where an error status is set and, then, control flows to step 1606, where the status is returned to the user interface. Otherwise, if the user has the privilege, control flows to step 1608. In step 1608, it is determined whether the locking of the qualification state definitions was successful. If not, control flows to step 1610, where an error status is set and, then, control flows to step 1606. Otherwise, if the locking was successful, control flows to step 1612. In step 1612, it is determined whether the new name of the qualification state being modified is unique, i.e., not used by any other qualification states. If not, control flows to step 1614, where an error status is set and, then, control flows to step 1615, where the qualification state definitions are unlocked. Otherwise, if the name is unique, control flows to step 1616. In step 1616, it is determined whether a valid fallback qualification state is specified. If not, control flows to step 1618, where an error status is set and, then, control flows to step 1615. Otherwise, if a valid fallback qualification state is specified, control flows to step 1620. In step 1620, it is determined whether a valid restricted signing qualification state is specified. If not, control flows to step 1622, where

an error status is set and, then, control flows to step 1615. Otherwise, if a valid restricted signing state is specified, control flows to step 1624. In step 1624, it is determined whether the qualification state was successfully updated in the database. If not, control flows to step 1626, where an error status is set and, then, control flows to step 1615. Otherwise, if the qualification state was successfully updated, then control flows to step 1615.

[0059] FIG. 17 shows an example method of processing the deleting of a qualification state. In step 1700, receiving a request to delete a qualification state definition from a user through a user interface. In step 1702, it is determined whether the user has the privilege to update the qualification state definitions. If not, control flows to step 1704, where an error status is set and, then, control flows to step 1706, where the status is returned to the user interface. Otherwise, if the user has the privilege, control flows to step 1708. In step 1708, it is determined whether the locking of the qualification state definitions was successful. If not, control flows to step 1710, where an error status is set and, then, control flows to step 1706. Otherwise, if the locking was successful, control flows to step 1712. In step 1712, it is determined whether the qualification state being deleted is referenced by any other qualification states as a fallback state, or a restricted signing state. If so, control flows to step 1714, where an error status is set and, then, control flows to step 1716, where the qualification state definitions are unlocked. Otherwise, if the qualification state being deleted is not referenced, then control flows to step 1718. In step 1718, it is determined whether there are any object versions in the source control system that are currently in the qualification state being deleted. If so, control flows to step 1720, where an error status is set and, then, control flows to step 1716. Otherwise, if no objects are in the qualification state being deleted, control flows to step 1722. In step 1722, it is determined whether the qualification state was successfully deleted from the database. If not, control flows to step 1724, where an error status is set and, then,

control flows to step 1716. Otherwise, if the deletion was successful, control flows to step 1716.

[0060] FIG. 18 shows an example system architecture for a source control system having an enforceable user-defined life cycle. A client/server PC 1800 is coupled via a network 1802 to a controller 1804 which communicates with various devices to provide process control. The client/server PC 1800 acts as both a client and a server so that a user has access to the source control system and the ability to command a load. Control strategies in a source control system are loaded from client/server 1800 to controller 1804, once they are in the appropriate life cycle state.

[0061] FIG. 19 shows another example system architecture for a source control system having an enforceable user-defined life cycle. Client PCs 1900 are used by operators to control a plant. Client PCs 1900 communicate with redundant server PCs 1902. Redundant server PCs 1902 are data engines or servers that provide data from the plant to client PCs 1900. Redundant server PCs 1902 are an alternative for greater availability than the single server PC 1800 of FIG. 18. When redundant server PCs 1902 are used, a back-up server takes over if a primary server fails. Control strategies in a source control system are loaded by PC 1900 or server 1902 to a controller 1904 over a network 1906, once they are in the appropriate life cycle state. Controllers 1904 communicate with various devices in a process control system.

[0062] It is to be understood that the above description is intended to be illustrative and not restrictive. Many other embodiments will be apparent to those of skill in the art upon reviewing the above description, such as adaptations of the present disclosure to control system configuration for applications other than process control systems. Various designs using hardware, software, and firmware

are contemplated by the present disclosure, even though some minor elements would need to change to better support the environments common to such systems and methods. The present disclosure has applicability to fields outside process control, such as software development environments and other kinds of systems needing control system configuration. Therefore, the scope of the present disclosure should be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.